ANEXO — Plano de implantacao em producao do Pro-Mental

Documento operacional complementar ao Trabalho de Conclusao de Curso.
Descreve como publicar a aplicacao web Pro-Mental em ambiente acessivel aos usuarios finais. **Nao contem credenciais de producao** — senhas, chaves e tokens sao configurados localmente em arquivos `.env`, excluidos do controle de versao.

A.1 Objetivo

Este anexo reune o roteiro de publicacao do sistema Pro-Mental, alinhado ao diagrama de implantacao apresentado na monografia (Figura 11) e aos artefatos versionados no repositorio do projeto. O objetivo e permitir que administradores tecnicos reproduzam o ambiente de producao com separacao entre interface, API e persistencia de dados.

A.2 Arquitetura adotada

A implantacao utiliza **dois nos de hospedagem**:

No	Provedor	Componente	Endereco publico
Front-end	Hostinger	Aplicacao Next.js	`https://promental.com.br`
Back-end e banco	VM Linux (projeto Pro-Mental)	API Laravel e MySQL em Docker	`https://api.promental.com.br`

Fluxo de comunicacao:

1. O navegador do usuario acessa a interface por HTTPS na Hostinger (`promental.com.br`).
2. O Next.js atende a interface e encaminha as chamadas REST ao backend por **proxy same-origin** (`/api/backend/...`), sem expor a URL da API no codigo JavaScript do browser.
3. O servidor Next.js (Hostinger) repassa as requisicoes por HTTPS ao subdominio `api`, na VM.
4. Na VM, o Nginx termina o TLS e encaminha ao container da API Laravel.
5. A autenticacao usa **cookie HttpOnly** (`promental_token`), definido pela API na resposta de login e reenviado automaticamente pelo browser nas requisicoes ao proxy.
6. A API persiste dados no MySQL, em container na mesma maquina, por conexao SQL restrita a rede interna do Docker.

Embora interface, API e banco permaneçam **logicamente separados**, o MySQL **nao e exposto** diretamente a internet. Chamadas diretas ao dominio `api.promental.com.br` (ex.: health check,

ferramentas de API) permanecem possiveis, com **CORS restrito** ao dominio do front-end.

A.3 Requisitos previos

A.3.1 Dominio e DNS

Dominio de producao: **promental.com.br**.

Registro DNS	Destino
`@` (raiz) e/ou `www`	Hostinger (front-end)
`api`	Endereco IP publico da VM Pro-Mental

A.3.2 Infraestrutura

- **Hostinger:** plano com suporte a Node.js (necessario ao Next.js em modo servidor).
- **VM:** Ubuntu 22.04 ou 24.04 LTS; configuracao minima sugerida de 2 vCPU, 4 GB de RAM e 80 GB de armazenamento SSD.
- **Certificados TLS:** Let's Encrypt (Certbot) na VM; SSL gerenciado no painel da Hostinger para o dominio principal.
- **E-mail transacional:** servico SMTP para recuperacao de senha e envio de credenciais a novos usuarios.

A.3.3 Software na VM

Docker Engine, Docker Compose, Nginx, Git e Certbot. Nao e necessaria instalacao nativa de PHP, Node.js ou MySQL no sistema host — esses componentes executam nos containers ou na Hostinger.

A.4 Configuracao da maquina virtual (API e banco)

A.4.1 Preparacao inicial

Apos acesso SSH, recomenda-se: atualizacao de pacotes, fuso horario `America/Sao_Paulo`, usuario dedicado de deploy, firewall liberando portas 22, 80 e 443, autenticacao SSH por chave e, opcionalmente, fail2ban.

A.4.2 Stack Docker

O repositorio inclui `docker-compose.yml` na raiz, com os servicos:

Servico	Funcao
`mysql`	Banco de dados MySQL 8.4 com volume persistente
`api`	API Laravel (PHP 8.3, Nginx e PHP-FPM em container)
`queue`	Processamento assincrono de filas
`scheduler`	Tarefas agendadas (`schedule:work`)

A API escuta apenas em `127.0.0.1:8080` no host; o acesso externo ocorre via Nginx com HTTPS.

A.4.3 Variaveis de ambiente

Templates versionados (sem segredos reais):

- `.env.docker.example` — variaveis do Docker Compose na raiz do repositorio;
- `backend/.env.production.example` — configuracao da aplicacao Laravel;
- `frontend/.env.production.example` — URLs consumidas pelo Next.js em **build/runtime no servidor** (nao use prefixo `NEXT_PUBLIC_` para a URL da API).

Principais correspondencias em producao:

Camada	Variavel	Valor
API	`APP_URL`	`https://api.promental.com.br`
API	`FRONTEND_URL`	`https://promental.com.br`
API	`APP_DEBUG`	`false`
API	`APP_KEY`	gerada com `php artisan key:generate`
API	`DB_HOST`	`mysql` (nome do servico Docker)
API	`AUTH_COOKIE_SECURE`	`true` (HTTPS obrigatorio)
API	`AUTH_COOKIE_SAME_SITE`	`lax`

API	`AUTH_COOKIE_DOMAIN`	vazio (cookie no host do front via proxy)
API	`SANCTUM_TOKEN_EXPIRATION`	`10080` (7 dias, em minutos)
API	`CORS_ALLOWED_ORIGINS`	`https://promental.com.br,https://www.pr`
API	`MAIL_MAILER` / `MAIL_*`	SMTP do provedor transacional
Front-end	`API_URL`	`https://api.promental.com.br` (somente servidor Next.js)
Front-end	`APP_URL`	`https://promental.com.br`

A URL em `FRONTEND_URL` deve coincidir exatamente com a origem que o usuario ve no navegador (incluindo uso ou nao de `www`), por causa de **CORS** (`config/cors.php`) e **cookies de autenticacao** (`config/auth_cookie.php`). Configuracoes publicadas: `backend/config/sanctum.php`, `backend/config/cors.php`.

A.4.4 Procedimento de deploy inicial na VM

1. Clonar o repositorio em `/var/www/promental`.
2. Copiar os templates de ambiente para `.env` (raiz) e `backend/.env`, preenchendo senhas fortes.
3. Executar o script `scripts/docker-up.sh`, que realiza build das imagens, inicializacao dos containers, geracao de chave da aplicacao, migracoes do banco e caches de producao.
4. Instalar a configuracao Nginx do host com `scripts/install-nginx-promental.sh`.
5. Emitir certificado TLS com Certbot para `api.promental.com.br`.
6. Validar o endpoint de saude da API (`/up`).

Seed inicial do banco (usuario administrador master):

1. Preencher `ADMIN_EMAIL`, `ADMIN_PASSWORD` e `ADMIN_CPF` no `backend/.env`.
2. Executar: `./scripts/seed-admin-producao.sh` ou `docker compose exec api php artisan db:seed --class=AdminMasterProducaoSeeder --force`.

Nao executar `db:seed` completo em producao (inclui dados de teste).

A.5 Publicacao do front-end na Hostinger

O front-end **nao e hospedado na VM**. O procedimento resumido e:

1. Definir `API_URL` e `APP_URL` no painel da Hostinger **antes** do build (variaveis de servidor; a URL da API nao vai para o bundle do browser).
2. Instalar dependencias (`pnpm install`) e gerar build de producao (`pnpm build`).
3. Publicar na Hostinger via aplicacao Node.js (Git, SFTP ou painel hPanel).
4. Comando de execucao: `pnpm start` (porta definida pelo provedor).
5. Ativar SSL para `promental.com.br` e configurar redirecionamento canonico (`www` ou apex).
6. Validar login (cookie HttpOnly), navegacao autenticada e fluxo CRAFFT apos o deploy.

O Next.js expoe a rota `frontend/src/app/api/backend/[...path]/route.ts`, que faz proxy das requisicoes do browser para `API_URL`, mantendo cookies de sessao no dominio do front-end.

Detalhamento adicional consta no repositorio, arquivo `docs/DEPLOY-FRONTEND-HOSTINGER.md`.

A.6 Rotinas operacionais

A.6.1 Backups

Backup diario do banco via script `scripts/backup_promental_db.sh`, agendado em cron. Retencao sugerida de 15 dias. Restauracao deve ser testada periodicamente em ambiente de homologacao. Backup completo e obrigatorio antes de migracoes de esquema em producao.

A.6.2 Atualizacoes

- **API:** script `scripts/docker-deploy.sh` (atualizacao de codigo, rebuild, migracoes e caches).
- **Front-end:** novo build na Hostinger e reinicio do processo Node.

A.6.3 Monitoramento minimo

Disponibilidade de `promental.com.br` e `api.promental.com.br`, uso de recursos da VM, estado dos containers Docker, logs da aplicacao e funcionamento da fila de processamento.

A.6.4 Rollback

Em falha de deploy da API: reverter para commit ou tag anterior, reconstruir containers e, se necessario, restaurar backup do banco. Em falha do front-end: reverter build na Hostinger e reiniciar o servico Node.

A.7 Checklist de go-live

- DNS propagado (`@`/`www` → Hostinger; `api` → VM).
- HTTPS ativo nos dois dominios publicos.
- `APP_DEBUG=false` e `APP_KEY` gerada na API.

- `FRONTEND_URL`, `API_URL` e `APP_URL` alinhados (mesma origem canonica, com ou sem `www`).
- `AUTH_COOKIE_SECURE=true` e `SANCTUM_TOKEN_EXPIRATION` definido.
- `CORS_ALLOWED_ORIGINS` restrito ao dominio do front-end.
- SMTP configurado e e-mail de recuperacao de senha testado.
- Containers Docker estaveis e migracoes aplicadas.
- Admin master criado (`./scripts/seed-admin-producao.sh`).
- Backups automaticos configurados e testados.
- Fluxos criticos testados: login, entrevista, termos, QDSD, dashboards e link CRAFFT.

A.8 Seguranca e divulgacao deste documento

Este plano descreve **procedimentos e arquitetura**, nao credenciais. Arquivos `.env` de producao permanecem fora do repositorio. A divulgacao deste anexo em formato PDF nao expoe segredos, desde que nao se incluam copias de configuracao preenchidas com dados reais.

Medidas recomendadas: senhas fortes, SSH por chave, firewall ativo, MySQL sem exposicao publica, autenticacao por cookie HttpOnly (sem token em `localStorage`), CORS restrito ao dominio do front-end, expiracao de tokens Sanctum e rotacao periodica de segredos.

A.9 Evolucao prevista

Integracao continua (CI/CD), cache e filas com Redis, banco gerenciado externo, ambiente de homologacao (staging) e infraestrutura como codigo (Terraform ou Ansible).

A.10 Referencia ao repositorio

Artefatos citados neste anexo encontram-se no repositorio do projeto Pro-Mental, pasta `docs/` e raiz do monorepo (`docker-compose.yml`, `scripts/`, `backend/Dockerfile`).

Repositorio: Tratar com autora do projeto
Plano operacional (Markdown): `docs/PLANO-IMPLANTACAO-PRODUCAO.md`
Deploy do front-end: `docs/DEPLOY-FRONTEND-HOSTINGER.md`

Elaborado como documentacao de implantacao do sistema Pro-Mental — UFES/CEUNES.